NAME

gpg2 - OpenPGP encryption and signing tool

SYNOPSIS

gpg2 [**--homedir** *dir*] [**--options** *file*] [*options*] *command* [*args*]

DESCRIPTION

gpg2 is the OpenPGP part of the GNU Privacy Guard (GnuPG). It is a tool to provide digital encryption and signing services using the OpenPGP standard. **gpg2** features complete key management and all bells and whistles you can expect from a decent OpenPGP implementation.

In contrast to the standalone version **gpg**, which is more suited for server and embedded platforms, this version is commonly installed under the name **gpg2** and more targeted to the desktop as it requires several other modules to be installed. The standalone version will be kept maintained and it is possible to install both versions on the same system. If you need to use different configuration files, you should make use of something like '*gpg.conf-2*' instead of just '*gpg.conf*'.

RETURN VALUE

The program returns 0 if everything was fine, 1 if at least a signature was bad, and other error codes for fatal errors.

WARNINGS

Use a *good* password for your user account and a *good* passphrase to protect your secret key. This passphrase is the weakest part of the whole system. Programs to do dictionary attacks on your secret keyring are very easy to write and so you should protect your ~/.gnupg/ directory very well.

Keep in mind that, if this program is used over a network (telnet), it is *very* easy to spy out your passphrase!

If you are going to verify detached signatures, make sure that the program knows about it; either give both filenames on the command line or use - to specify STDIN.

INTEROPERABILITY

GnuPG tries to be a very flexible implementation of the OpenPGP standard. In particular, GnuPG implements many of the optional parts of the standard, such as the SHA-512 hash, and the ZLIB and BZIP2 compression algorithms. It is important to be aware that not all OpenPGP programs implement these optional algorithms and that by forcing their use via the **--cipher-algo**, **--digest-algo**, **--cert-digest-algo**, or **--compress-algo** options in GnuPG, it is possible to create a perfectly valid OpenPGP message, but one that cannot be read by the intended recipient.

There are dozens of variations of OpenPGP programs available, and each supports a slightly different subset of these optional algorithms. For example, until recently, no (unhacked) version of PGP supported the BLOWFISH cipher algorithm. A message using BLOWFISH simply could not be read by a PGP user. By default, GnuPG uses the standard OpenPGP preferences system that will always do the right thing and create messages that are usable by all recipients, regardless of which OpenPGP program they use. Only override this safe default if you really know what you are doing.

If you absolutely must override the safe default, or if the preferences on a given key are invalid for some reason, you are far better off using the **--pgp6**, **--pgp7**, or **--pgp8** options. These options are safe as they do not force any particular algorithms in violation of OpenPGP, but rather reduce the available algorithms to a PGP-safe list.

COMMANDS

Commands are not distinguished from options except for the fact that only one command is allowed.

gpg2 may be run with no commands, in which case it will perform a reasonable action depending on the type of file it is given as input (an encrypted message is decrypted, a signature is verified, a file containing keys is listed).

Please remember that option as well as command parsing stops as soon as a non-option is encountered, you can explicitly stop parsing by using the special option **--**.

Commands not specific to the function

--version

Print the program version and licensing information. Note that you cannot abbreviate this command.

--help

-h Print a usage message summarizing the most useful command line options. Note that you cannot abbreviate this command.

--warranty

Print warranty information.

--dump-options

Print a list of all available options and commands. Note that you cannot abbreviate this command.

Commands to select the type of operation

--sign

-s Make a signature. This command may be combined with **--encrypt** (for a signed and encrypted message), **--symmetric** (for a signed and symmetrically encrypted message), or **--encrypt** and **--symmetric** together (for a signed message that may be decrypted via a secret key or a passphrase). The key to be used for signing is chosen by default or can be set with the **--local-user** and **--default-key** options.

--clearsign

Make a clear text signature. The content in a clear text signature is readable without any special software. OpenPGP software is only needed to verify the signature. Clear text signatures may modify end-of-line whitespace for platform independence and are not intended to be reversible. The key to be used for signing is chosen by default or can be set with the **--local-user** and **--default-key** options.

--detach-sign

-b Make a detached signature.

--encrypt

-e Encrypt data. This option may be combined with **--sign** (for a signed and encrypted message), **--symmetric** (for a message that may be decrypted via a secret key or a passphrase), or **--sign** and **--symmetric** together (for a signed message that may be decrypted via a secret key or a passphrase).

--symmetric

-c Encrypt with a symmetric cipher using a passphrase. The default symmetric cipher used is CAST5, but may be chosen with the **--cipher-algo** option. This option may be combined with **--sign** (for a signed and symmetrically encrypted message), **--encrypt** (for a message that may be decrypted via a secret key or a passphrase), or **--sign** and **--encrypt** together (for a signed message that may be decrypted via a secret key or a passphrase).

--store
Store only (make a simple RFC1991 literal data packet).

--decrypt

-d Decrypt the file given on the command line (or STDIN if no file is specified) and write it to STDOUT (or the file specified with **--output**). If the decrypted file is signed, the signature is also verified. This command differs from the default operation, as it never writes to the filename which is included in the file and it rejects files which don't begin with an encrypted message.

--verify
Assume that the first argument is a signed file or a detached signature and verify it without generating any output. With no arguments, the signature packet is read from STDIN. If only a sigfile is given, it may be a complete signature or a detached signature, in which case the signed stuff is expected in a file without the .sig or .asc extension. With more than 1 argument, the first should be a detached signature and the remaining files are the signed stuff. To read the signed stuff from STDIN, use - as the second filename. For security reasons a detached signature cannot read the signed material from STDIN without denoting it in the above way.

Note: When verifying a cleartext signature, **gpg** verifies only what makes up the cleartext signed data and not any extra data outside of the cleartext signature or header lines following directly the dash marker line. The option **--output** may be used to write out the actual signed data; but there are other pitfalls with this format as well. It is suggested to avoid cleartext signatures in favor of detached signatures.

--multifile
This modifies certain other commands to accept multiple files for processing on the command line or read from STDIN with each filename on a separate line. This allows for many files to be processed at once. **--multifile** may currently be used along with **--verify**, **--encrypt**, and **--decrypt**. Note that **--multifile --verify** may not be used with detached signatures.

--verify-files
Identical to **--multifile --verify**.

--encrypt-files
Identical to **--multifile --encrypt**.

--decrypt-files
Identical to **--multifile --decrypt**.

--list-keys

-k

--list-public-keys
List all keys from the public keyrings, or just the keys given on the command line.

Avoid using the output of this command in scripts or other programs as it is likely to change as GnuPG changes. See **--with-colons** for a machine-parseable key listing command that is appropriate for use in scripts and other programs.

--list-secret-keys

-K List all keys from the secret keyrings, or just the ones given on the command line. A # after the letters **sec** means that the secret key is not usable (for example, if it was created via **--export-secret-subkeys**).

--list-sigs
Same as **--list-keys**, but the signatures are listed too. This command has the same effect as using **--list-keys** with **--with-sig-list**.

For each signature listed, there are several flags in between the sig tag and keyid. These flags give additional information about each signature. From left to right, they are the numbers 1-3 for certificate check level (see **--ask-cert-level**), L for a local or non-exportable signature (see **--lsign-key**), R for a nonRevocable signature (see the **--edit-key** command nrsign), P for a signature that contains a policy URL (see **--cert-policy-url**), N for a signature that contains a notation (see **--cert-notation**), X for an eXpired signature (see **--ask-cert-expire**), and the numbers 1-9 or T for 10 and above to indicate trust signature levels (see the **--edit-key** command tsign).

--check-sigs
Same as **--list-sigs**, but the signatures are verified. Note that for performance reasons the revocation status of a signing key is not shown. This command has the same effect as using **--list-keys** with **--with-sig-check**.

The status of the verification is indicated by a flag directly following the sig tag (and thus before the flags described above for **--list-sigs**). A ! indicates that the signature has been successfully verified, a - denotes a bad signature and a % is used if an error occurred while checking the signature (e.g. a non supported algorithm).

--locate-keys
Locate the keys given as arguments. This command basically uses the same algorithm as used when locating keys for encryption or signing and may thus be used to see what keys **gpg2** might use. In particular external methods as defined by **--auto-key-locate** may be used to locate a key. Only public keys are listed.

--fingerprint
List all keys (or the specified ones) along with their fingerprints. This is the same output as **--list-keys** but with the additional output of a line with the fingerprint. May also be combined with **--list-sigs** or **--check-sigs**. If this command is given twice, the fingerprints of all secondary keys are listed too.

--list-packets
List only the sequence of packets. This is mainly useful for debugging.

--card-edit
Present a menu to work with a smartcard. The subcommand help provides an overview on available commands. For a detailed description, please see the Card HOWTO at https://gnupg.org/documentation/howtos.html#GnuPG-cardHOWTO

--card-status
Show the content of the smart card.

--change-pin
Present a menu to allow changing the PIN of a smartcard. This functionality is also available as the subcommand passwd with the **--card-edit** command.

--delete-key name
Remove key from the public keyring. In batch mode either **--yes** is required or the key must be specified by fingerprint. This is a safeguard against accidental deletion of multiple keys.

--delete-secret-key name
Remove key from the secret keyring. In batch mode the key must be specified by fingerprint.

--delete-secret-and-public-key name
Same as **--delete-key**, but if a secret key exists, it will be removed first. In batch mode the key must be specified by fingerprint.

--export
Either export all keys from all keyrings (default keyrings and those registered via option **--keyring**), or if at least one name is given, those of the given name. The exported keys are written to STDOUT or to the file given with option **--output**. Use together with **--armor** to mail those keys.

--send-keys key IDs
Similar to **--export** but sends the keys to a keyserver. Fingerprints may be used instead of key IDs. Option **--keyserver** must be used to give the name of this keyserver. Don't send your complete keyring to a keyserver --- select only those keys which are new or changed by you. If no key IDs are given, **gpg** does nothing.

--export-secret-keys

--export-secret-subkeys
Same as **--export**, but exports the secret keys instead. The exported keys are written to STDOUT or to the file given with option **--output**. This command is often used along with the option **--armor** to allow easy printing of the key for paper backup; however the external tool **paperkey** does a better job for creating backups on paper. Note that exporting a secret key can be a security risk if the exported keys are send over an insecure channel.

The second form of the command has the special property to render the secret part of the primary key useless; this is a GNU extension to OpenPGP and other implementations can not be expected to successfully import such a key. Its intended use is to generated a full key with an additional signing subkey on a dedicated machine and then using this command to export the key without the primary key to the main machine.

See the option **--simple-sk-checksum** if you want to import an exported secret key into ancient OpenPGP implementations.

--import

--fast-import
Import/merge keys. This adds the given keys to the keyring. The fast version is currently just a synonym.

There are a few other options which control how this command works. Most notable here is the **--import-options merge-only** option which does not insert new keys but does only the merging of new signatures, user-IDs and subkeys.

--recv-keys key IDs
Import the keys with the given key IDs from a keyserver. Option **--keyserver** must be used to give the name of this keyserver.

--refresh-keys
Request updates from a keyserver for keys that already exist on the local keyring. This is useful for updating a key with the latest signatures, user IDs, etc. Calling this with no arguments will refresh the entire keyring. Option **--keyserver** must be used to give the name of the keyserver for all keys that do not have preferred keyservers set (see

--keyserver-options honor-keyserver-url).

--search-keys names
Search the keyserver for the given names. Multiple names given here will be joined together to create the search string for the keyserver. Option**--k eyserver** must be used to give the name of this keyserver. Keyservers that support different search methods allow using the syntax specified in How to specify a user ID below. Note that different keyserver types support different search methods. Currently only LDAP supports them all.

--fetch-keys URIs
Retrieve keys located at the specified URIs. Note that different installations of GnuPG may support different protocols (HTTP, FTP, LDAP, etc.)

--update-trustdb
Do trust database maintenance. This command iterates over all keys and builds the Web of Trust. This is an interactive command because it may have to ask for the ownertrust values for keys. The user has to give an estimation of how far she trusts the owner of the displayed key to correctly certify (sign) other keys. GnuPG only asks for the ownertrust value if it has not yet been assigned to a key. Using the **--edit-key** menu, the assigned value can be changed at any time.

--check-trustdb
Do trust database maintenance without user interaction. From time to time the trust database must be updated so that expired keys or signatures and the resulting changes in the Web of Trust can be tracked. Normally, GnuPG will calculate when this is required and do it automatically unless **--no-auto-check-trustdb** is set. This command can be used to force a trust database check at any time. The processing is identical to that of **--update-trustdb** but it skips keys with a not yet defined ownertrust.

For use with cron jobs, this command can be used together with **--batch** in which case the trust database check is done only if a check is needed. To force a run even in batch mode add the option **--yes**.

--export-ownertrust
Send the ownertrust values to STDOUT. This is useful for backup purposes as these values are the only ones which can't be re-created from a corrupted trustdb. Example:
gpg2 --export-ownertrust > otrust.txt

--import-ownertrust
Update the trustdb with the ownertrust values stored in **files** (or STDIN if not given); existing values will be overwritten. In case of a severely damaged trustdb and if you have a recent backup of the ownertrust values (e.g. in the file '*otrust.txt*', you may re-create the trustdb using these commands:
cd ~/.gnupg
rm trustdb.gpg
gpg2 --import-ownertrust < otrust.txt

--rebuild-keydb-caches
When updating from version 1.0.6 to 1.0.7 this command should be used to create signature caches in the keyring. It might be handy in other situations too.

--print-md algo

--print-mds
Print message digest of algorithm ALGO for all given files or STDIN. With the second form (or a deprecated * as algo) digests for all available algorithms are printed.

--gen-random 0|1|2 count
Emit *count* random bytes of the given quality level 0, 1 or 2. If *count* is not given or zero, an endless sequence of random bytes will be emitted. If used with **--armor** the output will be base64 encoded. PLEASE, don't use this command unless you know what you are doing; it may remove precious entropy from the system!

--gen-prime mode bits
Use the source, Luke :-). The output format is still subject to change.

--enarmor

--dearmor
Pack or unpack an arbitrary input into/from an OpenPGP ASCII armor. This is a GnuPG extension to OpenPGP and in general not very useful.

How to manage your keys
This section explains the main commands for key management

--gen-key
Generate a new key pair. This command is normally only used interactively.

There is an experimental feature which allows you to create keys in batch mode. See the file '*doc/DETAILS*' in the source distribution on how to use this.

--gen-revoke name
Generate a revocation certificate for the complete key. To revoke a subkey or a signature, use the **--edit** command.

--desig-revoke name
Generate a designated revocation certificate for a key. This allows a user (with the permission of the keyholder) to revoke someone else's key.

--edit-key
Present a menu which enables you to do most of the key management related tasks. It expects the specification of a key on the command line.

uid n Toggle selection of user ID or photographic user ID with index **n**. Use***** to select all and **0** to deselect all.

key n Toggle selection of subkey with index **n**. Use***** to select all and **0** to deselect all.

sign Make a signature on key of user **name** If the key is not yet signed by the default user (or the users given with -u), the program displays the information of the key again, together with its fingerprint and asks whether it should be signed. This question is repeated for all users specified with -u.

lsign Same as sign but the signature is marked as non-exportable and will therefore never be used by others. This may be used to make keys valid only in the local environment.

nrsign Same as sign but the signature is marked as non-revocable and can therefore never be revoked.

tsign Make a trust signature. This is a signature that combines the notions of certification (like a regular signature), and trust (like the trust command). It is generally only useful in distinct communities or groups.

Note that l (for local / non-exportable), nr (for non-revocable, and t (for trust) may be freely mixed and prefixed to sign to create a signature of any type desired.

delsig Delete a signature. Note that it is not possible to retract a signature, once it has been send to the public (i.e. to a keyserver). In that case you better use **revsig**.

revsig Revoke a signature. For every signature which has been generated by one of the secret keys, GnuPG asks whether a revocation certificate should be generated.

check Check the signatures on all selected user IDs.

adduid
Create an additional user ID.

addphoto
Create a photographic user ID. This will prompt for a JPEG file that will be embedded into the user ID. Note that a very large JPEG will make for a very large key. Also note that some programs will display your JPEG unchanged (GnuPG), and some programs will scale it to fit in a dialog box (PGP).

showphoto
Display the selected photographic user ID.

deluid Delete a user ID or photographic user ID. Note that it is not possible to retract a user id, once it has been send to the public (i.e. to a keyserver). In that case you better use **revuid**.

revuid
Revoke a user ID or photographic user ID.

primary
Flag the current user id as the primary one, removes the primary user id flag from all other user ids and sets the timestamp of all affected self-signatures one second ahead. Note that setting a photo user ID as primary makes it primary over other photo user IDs, and setting a regular user ID as primary makes it primary over other regular user IDs.

keyserver
Set a preferred keyserver for the specified user ID(s). This allows other users to know where you prefer they get your key from. See **--keyserver-options honor-keyserver-url** for more on how this works. Setting a value of none removes an existing preferred keyserver.

notation
Set a name=value notation for the specified user ID(s). See **--cert-notation** for more on how this works. Setting a value of none removes all notations, setting a notation prefixed with a minus sign (-) removes that notation, and setting a notation name (without the =value) prefixed with a minus sign removes all notations with that name.

pref List preferences from the selected user ID. This shows the actual preferences, without including any implied preferences.

showpref
More verbose preferences listing for the selected user ID. This shows the preferences in effect by including the implied preferences of 3DES (cipher), SHA-1 (digest), and Uncompressed (compression) if they are not already included in the preference list. In addition, the preferred keyserver and signature notations (if any) are shown.

setpref string
Set the list of user ID preferences to **string** for all (or just the selected) user IDs. Calling setpref with no arguments sets the preference list to the default (either built-in or set via **--default-preference-list**), and calling setpref with none as the argument sets an empty preference list. Use **gpg2 --version** to get a list of available algorithms. Note that while you can change the preferences on an attribute user ID (aka photo ID), GnuPG does not select keys via attribute user IDs so these preferences will not be used by GnuPG.

When setting preferences, you should list the algorithms in the order which you'd like to see them used by someone else when encrypting a message to your key. If you don't include 3DES, it will be automatically added at the end. Note that there are many factors that go into choosing an algorithm (for example, your key may not be the only recipient), and so the remote OpenPGP application being used to send to you may or may not follow your exact chosen order for a given message. It will, however, only choose an algorithm that is present on the preference list of every recipient key. See also the INTEROPERABILITY WITH OTHER OPENPGP PROGRAMS section below.

addkey
Add a subkey to this key.

addcardkey
Generate a subkey on a card and add it to this key.

keytocard
Transfer the selected secret subkey (or the primary key if no subkey has been selected) to a smartcard. The secret key in the keyring will be replaced by a stub if the key could be stored successfully on the card and you use the save command later. Only certain key types may be transferred to the card. A sub menu allows you to select on what card to store the key. Note that it is not possible to get that key back from the card - if the card gets broken your secret key will be lost unless you have a backup somewhere.

bkuptocard file
Restore the given file to a card. This command may be used to restore a backup key (as generated during card initialization) to a new card. In almost all cases this will be the encryption key. You should use this command only with the corresponding public key and make sure that the file given as argument is indeed the backup to restore. You should then select 2 to restore as encryption key. You will first be asked to enter the passphrase of the backup key and then for the Admin PIN of the card.

delkey
Remove a subkey (secondart key). Note that it is not possible to retract a subkey, once it has been send to the public (i.e. to a keyserver). In that case you better use **revkey**.

revkey
Revoke a subkey.

expire Change the key or subkey expiration time. If a subkey is selected, the expiration time of this subkey will be changed. With no selection, the key expiration of the primary key is changed.

trust Change the owner trust value for the key. This updates the trust-db immediately and no save is required.

disable

enable
Disable or enable an entire key. A disabled key can not normally be used for encryption.

addrevoker
Add a designated revoker to the key. This takes one optional argument: sensitive. If a designated revoker is marked as sensitive, it will not be exported by default (see export-options).

passwd
Change the passphrase of the secret key.

toggle Toggle between public and secret key listing.

clean Compact (by removing all signatures except the selfsig) any user ID that is no longer usable (e.g. revoked, or expired). Then, remove any signatures that are not usable by the trust calculations. Specifically, this removes any signature that does not validate, any signature that is superseded by a later signature, revoked signatures, and signatures issued by keys that are not present on the keyring.

minimize
Make the key as small as possible. This removes all signatures from each user ID except for the most recent self-signature.

cross-certify
Add cross-certification signatures to signing subkeys that may not currently have them. Cross-certification signatures protect against a subtle attack against signing subkeys. See **--require-cross-certification**. All new keys generated have this signature by default, so this option is only useful to bring older keys up to date.

save Save all changes to the key rings and quit.

quit Quit the program without updating the key rings.

The listing shows you the key with its secondary keys and all user ids. The primary user id is indicated by a dot, and selected keys or user ids are indicated by an asterisk. The trust value is displayed with the primary key: the first is the assigned owner trust and the second is the calculated trust value. Letters are used for the values:

- No ownertrust assigned / not yet calculated.

e Trust calculation has failed; probably due to an expired key.

q Not enough information for calculation.

n Never trust this key.

m Marginally trusted.

f Fully trusted.

u Ultimately trusted.

--sign-key name
Signs a public key with your secret key. This is a shortcut version of the subcommand sign from **--edit**.

--lsign-key name
Signs a public key with your secret key but marks it as non-exportable. This is a shortcut version of the subcommand lsign from **--edit-key**.

--passwd *user_id*
Change the passphrase of the secret key belonging to the certificate specified as *user_id*. This is a shortcut for the sub-command **passwd** of the edit key menu.

OPTIONS

gpg2 features a bunch of options to control the exact behaviour and to change the default configuration.

Long options can be put in an options file (default ~/.gnupg/gpg.conf). Short option names will not work - for example, armor is a valid option for the options file, while a is not. Do not write the 2 dashes, but simply the name of the option and any required arguments. Lines with a hash ('#') as the first non-white-space character are ignored. Commands may be put in this file too, but that is not generally useful as the command will execute automatically with every execution of gpg.

Please remember that option parsing stops as soon as a non-option is encountered, you can explicitly stop parsing by using the special option **--**.

How to change the configuration

These options are used to change the configuration and are usually found in the option file.

--default-key *name*
Use *name* as the default key to sign with. If this option is not used, the default key is the first key found in the secret keyring. Note that**-u** or **--local-user** o verrides this option.

--default-recipient *name*
Use *name* as default recipient if option **--recipient** is not used and don't ask if this is a valid one. *name* must be non-empty.

--default-recipient-self
Use the default key as default recipient if option **--recipient** is not used and don't ask if this is a valid one. The default key is the first one from the secret keyring or the one set with **--default-key**.

--no-default-recipient
Reset **--default-recipient** and **--default-recipient-self**.

-v, --verbose
Give more information during processing. If used twice, the input data is listed in detail.

--no-verbose
Reset verbose level to 0.

-q, --quiet
Try to be as quiet as possible.

--batch

--no-batch
Use batch mode. Never ask, do not allow interactive commands. **--no-batc h** disables this option. Note that even with a filename given on the command line, gpg might still need to read from STDIN (in particular if gpg figures that the input is a detached signature and no data file has been specified). Thus if you do not want to feed data via STDIN, you should connect STDIN to '*/dev/null*'.

--no-tty
Make sure that the TTY (terminal) is never used for any output. This option is needed in some cases because GnuPG sometimes prints warnings to the TTY even if **--batch** is used.

--yes Assume yes on most questions.

--no Assume no on most questions.

--list-options parameters
This is a space or comma delimited string that gives options used when listing keys and signatures (that is, **--list-keys**, **--list-sigs**, **--list-public-keys**, **--list-secret-keys**, and the **--edit-key** functions). Options can be prepended with a **no-** (after the two dashes) to give the opposite meaning. The options are:

show-photos
Causes **--list-keys**, **--list-sigs**, **--list-public-keys**, and **--list-secret-keys** to display any photo IDs attached to the key. Defaults to no. See also **--photo-viewer**. Does not work with **--with-colons**: see **--attribute-fd** for the appropriate way to get photo data for scripts and other frontends.

show-policy-urls
Show policy URLs in the **--list-sigs** or **--check-sigs** listings. Defaults to no.

show-notations

show-std-notations

show-user-notations
Show all, IETF standard, or user-defined signature notations in the **--list-sigs** or **--check-sigs** listings. Defaults to no.

show-keyserver-urls
Show any preferred keyserver URL in the **--list-sigs** or **--check-sigs** listings. Defaults to no.

show-uid-validity
Display the calculated validity of user IDs during key listings. Defaults to no.

show-unusable-uids
Show revoked and expired user IDs in key listings. Defaults to no.

show-unusable-subkeys
Show revoked and expired subkeys in key listings. Defaults to no.

show-keyring
Display the keyring name at the head of key listings to show which keyring a given key resides on. Defaults to no.

show-sig-expire
Show signature expiration dates (if any) during **--list-sigs** or **--check-sigs** listings. Defaults to no.

show-sig-subpackets
Include signature subpackets in the key listing. This option can take an optional argument list of the subpackets to list. If no argument is passed, list all subpackets. Defaults to no. This option is only meaningful when using **--with-colons** along with **--list-sigs** or **--check-sigs**.

--verify-options parameters
This is a space or comma delimited string that gives options used when verifying signatures. Options can be prepended with a 'no-' to give the opposite meaning. The options are:

show-photos
Display any photo IDs present on the key that issued the signature. Defaults to no. See also **--photo-viewer**.

show-policy-urls
Show policy URLs in the signature being verified. Defaults to no.

show-notations

show-std-notations

show-user-notations
Show all, IETF standard, or user-defined signature notations in the signature being verified. Defaults to IETF standard.

show-keyserver-urls
Show any preferred keyserver URL in the signature being verified. Defaults to no.

show-uid-validity
Display the calculated validity of the user IDs on the key that issued the signature. Defaults to no.

show-unusable-uids
Show revoked and expired user IDs during signature verification. Defaults to no.

show-primary-uid-only
Show only the primary user ID during signature verification. That is all the AKA lines as well as photo Ids are not shown with the signature verification status.

pka-lookups
Enable PKA lookups to verify sender addresses. Note that PKA is based on DNS, and so enabling this option may disclose information on when and what signatures are verified or to whom data is encrypted. This is similar to the web bug described for the auto-key-retrieve feature.

pka-trust-increase
Raise the trust in a signature to full if the signature passes PKA validation. This option is only meaningful if pka-lookups is set.

--enable-large-rsa

--disable-large-rsa
With --gen-key and --batch, enable the creation of larger RSA secret keys than is generally recommended (up to 8192 bits). These large keys are more expensive to use, and their signatures and certifications are also larger.

--enable-dsa2

--disable-dsa2
Enable hash truncation for all DSA keys even for old DSA Keys up to 1024 bit. This is also the default with **--openpgp**. Note that older versions of GnuPG also required this flag to allow the generation of DSA larger than 1024 bit.

--photo-viewer string
This is the command line that should be run to view a photo ID. %i will be expanded to a filename containing the photo. %I does the same, except the file will not be deleted once the viewer exits. Other flags are %k for the key ID, %K for the long key ID, %f for the key fingerprint, %t for the extension of the image type (e.g. jpg), %T for the MIME type of the image (e.g. image/jpeg), %v for the single-character calculated validity of the image being viewed (e.g. f), %V for the calculated validity as a string (e.g. full), %U for a base32 encoded hash of the user ID, and %% for an actual percent sign. If neither %i or %I are present, then the photo will be supplied to the viewer on standard input.

The default viewer is xloadimage -fork -quiet -title 'KeyID 0x%k' STDIN. Note that if your image viewer program is not secure, then executing it from GnuPG does not make it secure.

--exec-path string
Sets a list of directories to search for photo viewers and keyserver helpers. If not provided, keyserver helpers use the compiled-in default directory, and photo viewers use the $PATH environment variable. Note, that on W32 system this value is ignored when searching for keyserver helpers.

--keyring file
Add **file** to the current list of keyrings. If **file** begins with a tilde and a slash, these are replaced by the $HOME directory. If the filename does not contain a slash, it is assumed to be in the GnuPG home directory (~/.gnupg if **--homedir** or $GNUPGHOME is not used).

Note that this adds a keyring to the current list. If the intent is to use the specified keyring alone, use **--keyring** along with **--no-default-keyring**.

--secret-keyring file
Same as **--keyring** but for the secret keyrings.

--primary-keyring file
Designate **file** as the primary public keyring. This means that newly imported keys (via **--import** or keyserver **--recv-from**) will go to this keyring.

--trustdb-name file
Use **file** instead of the default trustdb. If **file** begins with a tilde and a slash, these are replaced by the $HOME directory. If the filename does not contain a slash, it is assumed to be in the GnuPG home directory (‘*~/.gnupg*’ if**--homedir** or $GNUPGHOME is not used).

--homedir *dir*
Set the name of the home directory to *dir*. If this option is not used, the home directory defaults to ‘*~/.gnupg*’. It is only recognized when given on the command line. It also overrides any home directory stated through the environment variable ‘*GNUPGHOME*’ or (on W32 systems) by means of the Registry entry *HKCUSoftwareGNUGnuPG:HomeDir*.

--display-charset name
Set the name of the native character set. This is used to convert some informational strings like user IDs to the proper UTF-8 encoding. Note that this has nothing to do with the character set of data to be encrypted or signed; GnuPG does not recode user-supplied data. If this option is not used, the default character set is determined from the current locale. A verbosity level of 3 shows the chosen set. Valid values for **name** are:

iso-8859-1
This is the Latin 1 set.

iso-8859-2
The Latin 2 set.

iso-8859-15
This is currently an alias for the Latin 1 set.

koi8-r The usual Russian set (rfc1489).

utf-8 Bypass all translations and assume that the OS uses native UTF-8 encoding.

--utf8-strings

--no-utf8-strings
Assume that command line arguments are given as UTF8 strings. The default (**--no-utf8-strings**) is to assume that arguments are encoded in the character set as specified by **--display-charset**. These options affect all following arguments. Both options may be used multiple times.

--options file
Read options from **file** and do not try to read them from the default options file in the homedir (see **--homedir**). This option is ignored if used in an options file.

--no-options
Shortcut for **--options /dev/null**. This option is detected before an attempt to open an option file. Using this option will also prevent the creation of a ‘*~/.gnupg*’ homedir.

-z n

--compress-level n

--bzip2-compress-level n

Set compression level to **n** for the ZIP and ZLIB compression algorithms. The default is to use the default compression level of zlib (normally 6). **--bzip2-compress-level** sets the compression level for the BZIP2 compression algorithm (defaulting to 6 as well). This is a different option from **--compress-level** since BZIP2 uses a significant amount of memory for each additional compression level. **-z** sets both. A value of 0 for **n** disables compression.

--bzip2-decompress-lowmem

Use a different decompression method for BZIP2 compressed files. This alternate method uses a bit more than half the memory, but also runs at half the speed. This is useful under extreme low memory circumstances when the file was originally compressed at a high **--bzip2-compress-level**.

--mangle-dos-filenames

--no-mangle-dos-filenames

Older version of Windows cannot handle filenames with more than one dot. **--mangle-dos-filenames** causes GnuPG to replace (rather than add to) the extension of an output filename to avoid this problem. This option is off by default and has no effect on non-Windows platforms.

--ask-cert-level

--no-ask-cert-level

When making a key signature, prompt for a certification level. If this option is not specified, the certification level used is set via **--default-cert-level**. See **--default-cert-level** for information on the specific levels and how they are used. **--no-ask-cert-level** disables this option. This option defaults to no.

--default-cert-level n

The default to use for the check level when signing a key.

0 means you make no particular claim as to how carefully you verified the key.

1 means you believe the key is owned by the person who claims to own it but you could not, or did not verify the key at all. This is useful for a persona verification, where you sign the key of a pseudonymous user.

2 means you did casual verification of the key. For example, this could mean that you verified the key fingerprint and checked the user ID on the key against a photo ID.

3 means you did extensive verification of the key. For example, this could mean that you verified the key fingerprint with the owner of the key in person, and that you checked, by means of a hard to forge document with a photo ID (such as a passport) that the name of the key owner matches the name in the user ID on the key, and finally that you verified (by exchange of email) that the email address on the key belongs to the key owner.

Note that the examples given above for levels 2 and 3 are just that: examples. In the end, it is up to you to decide just what casual and extensive mean to you.

This option defaults to 0 (no particular claim).

--min-cert-level

When building the trust database, treat any signatures with a certification level below this as invalid. Defaults to 2, which disregards level 1 signatures. Note that level 0 no particular claim signatures are always accepted.

--trusted-key long key ID
Assume that the specified key (which must be given as a full 8 byte key ID) is as trustworthy as one of your own secret keys. This option is useful if you don't want to keep your secret keys (or one of them) online but still want to be able to check the validity of a given recipient's or signator's key.

--trust-model pgp|classic|direct|always|auto
Set what trust model GnuPG should follow. The models are:

pgp This is the Web of Trust combined with trust signatures as used in PGP 5.x and later. This is the default trust model when creating a new trust database.

classic
This is the standard Web of Trust as used in PGP 2.x and earlier.

direct Key validity is set directly by the user and not calculated via the Web of Trust.

always
Skip key validation and assume that used keys are always fully trusted. You generally won't use this unless you are using some external validation scheme. This option also suppresses the [uncertain] tag printed with signature checks when there is no evidence that the user ID is bound to the key. Note that this trust model still does not allow the use of expired, revoked, or disabled keys.

auto Select the trust model depending on whatever the internal trust database says. This is the default model if such a database already exists.

--auto-key-locate parameters

--no-auto-key-locate
GnuPG can automatically locate and retrieve keys as needed using this option. This happens when encrypting to an email address (in the user@example.com form), and there are no user@example.com keys on the local keyring. This option takes any number of the following mechanisms, in the order they are to be tried:

cert Locate a key using DNS CERT, as specified in rfc4398.

pka Locate a key using DNS PKA.

ldap Using DNS Service Discovery, check the domain in question for any LDAP keyservers to use. If this fails, attempt to locate the key using the PGP Universal method of checking ldap://keys.(thedomain).

keyserver
Locate a key using whatever keyserver is defined using the **--keyserver** option.

keyserver-URL
In addition, a keyserver URL as used in the **--keyserver** option may be used here to query that particular keyserver.

local Locate the key using the local keyrings. This mechanism allows to select the order a local key lookup is done. Thus using --auto-key-locate local is identical to **--no-auto-key-locate**.

nodefault
This flag disables the standard local key lookup, done before any of the mechanisms defined by the **--auto-key-locate** are tried. The position of this mechanism in the list does not matter. It is not required if **local** is also used.

clear Clear all defined mechanisms. This is useful to override mechanisms given in a config file.

--keyid-format short|0xshort|long|0xlong
Select how to display key IDs. short is the traditional 8-character key ID. long is the more accurate (but less convenient) 16-character key ID. Add an 0x to either to include an 0x at the beginning of the key ID, as in 0x99242560. Note that this option is ignored if the option --with-colons is used.

--keyserver name
Use **name** as your keyserver. This is the server that **--recv-keys**, **--send-keys**, and **--search-keys** will communicate with to receive keys from, send keys to, and search for keys on. The format of the **name** is a URI: 'scheme:[//]keyservername[:port]' The scheme is the type of keyserver: hkp for the HTTP (or compatible) keyservers, ldap for the LDAP keyservers, or mailto for the Graff email keyserver. Note that your particular installation of GnuPG may have other keyserver types available as well. Keyserver schemes are case-insensitive. After the keyserver name, optional keyserver configuration options may be provided. These are the same as the global **--keyserver-options** from below, but apply only to this particular keyserver.

Most keyservers synchronize with each other, so there is generally no need to send keys to more than one server. The keyserver **hkp://keys.gnupg.net** uses round robin DNS to give a different keyserver each time you use it.

--keyserver-options name=value1
This is a space or comma delimited string that gives options for the keyserver. Options can be prefixed with a 'no-' to give the opposite meaning. Valid import-options or export-options may be used here as well to apply to importing (**--recv-key**) or exporting (**--send-key**) a key from a keyserver. While not all options are available for all keyserver types, some common options are:

include-revoked
When searching for a key with **--search-keys**, include keys that are marked on the keyserver as revoked. Note that not all keyservers differentiate between revoked and unrevoked keys, and for such keyservers this option is meaningless. Note also that most keyservers do not have cryptographic verification of key revocations, and so turning this option off may result in skipping keys that are incorrectly marked as revoked.

include-disabled
When searching for a key with **--search-keys**, include keys that are marked on the keyserver as disabled. Note that this option is not used with HKP keyservers.

auto-key-retrieve
This option enables the automatic retrieving of keys from a keyserver when verifying signatures made by keys that are not on the local keyring.

Note that this option makes a web bug like behavior possible. Keyserver operators can see which keys you request, so by sending you a message signed by a brand new key (which you naturally will not have on your local keyring), the

operator can tell both your IP address and the time when you verified the signature.

honor-keyserver-url
When using **--refresh-keys**, if the key in question has a preferred keyserver URL, then use that preferred keyserver to refresh the key from. In addition, if auto-key-retrieve is set, and the signature being verified has a preferred keyserver URL, then use that preferred keyserver to fetch the key from. Defaults to yes.

honor-pka-record
If auto-key-retrieve is set, and the signature being verified has a PKA record, then use the PKA information to fetch the key. Defaults to yes.

include-subkeys
When receiving a key, include subkeys as potential targets. Note that this option is not used with HKP keyservers, as they do not support retrieving keys by subkey id.

use-temp-files
On most Unix-like platforms, GnuPG communicates with the keyserver helper program via pipes, which is the most efficient method. This option forces GnuPG to use temporary files to communicate. On some platforms (such as Win32 and RISC OS), this option is always enabled.

keep-temp-files
If using 'use-temp-files', do not delete the temp files after using them. This option is useful to learn the keyserver communication protocol by reading the temporary files.

verbose
Tell the keyserver helper program to be more verbose. This option can be repeated multiple times to increase the verbosity level.

timeout
Tell the keyserver helper program how long (in seconds) to try and perform a keyserver action before giving up. Note that performing multiple actions at the same time uses this timeout value per action. For example, when retrieving multiple keys via **--recv-keys**, the timeout applies separately to each key retrieval, and not to the **--recv-keys** command as a whole. Defaults to 30 seconds.

http-proxy=value
Set the proxy to use for HTTP and HKP keyservers. This overrides the http_proxy environment variable, if any.

max-cert-size
When retrieving a key via DNS CERT, only accept keys up to this size. Defaults to 16384 bytes.

debug Turn on debug output in the keyserver helper program. Note that the details of debug output depends on which keyserver helper program is being used, and in turn, on any libraries that the keyserver helper program uses internally (libcurl, openldap, etc).

check-cert
Enable certificate checking if the keyserver presents one (for hkps or ldaps). Defaults to on.

ca-cert-file
Provide a certificate store to override the system default. Only necessary if check-cert is enabled, and the keyserver is using a certificate that is not present in a system default certificate list.

Note that depending on the SSL library that the keyserver helper is built with, this may actually be a directory or a file.

--completes-needed n
Number of completely trusted users to introduce a new key signer (defaults to 1).

--marginals-needed n
Number of marginally trusted users to introduce a new key signer (defaults to 3)

--max-cert-depth n
Maximum depth of a certification chain (default is 5).

--simple-sk-checksum
Secret keys are integrity protected by using a SHA-1 checksum. This method is part of the upcoming enhanced OpenPGP specification but GnuPG already uses it as a countermeasure against certain attacks. Old applications don't understand this new format, so this option may be used to switch back to the old behaviour. Using this option bears a security risk. Note that using this option only takes effect when the secret key is encrypted - the simplest way to make this happen is to change the passphrase on the key (even changing it to the same value is acceptable).

--no-sig-cache
Do not cache the verification status of key signatures. Caching gives a much better performance in key listings. However, if you suspect that your public keyring is not save against write modifications, you can use this option to disable the caching. It probably does not make sense to disable it because all kind of damage can be done if someone else has write access to your public keyring.

--no-sig-create-check
GnuPG normally verifies each signature right after creation to protect against bugs and hardware malfunctions which could leak out bits from the secret key. This extra verification needs some time (about 115% for DSA keys), and so this option can be used to disable it. However, due to the fact that the signature creation needs manual interaction, this performance penalty does not matter in most settings.

--auto-check-trustdb

--no-auto-check-trustdb
If GnuPG feels that its information about the Web of Trust has to be updated, it automatically runs the **--check-trustdb** command internally. This may be a time consuming process. **--no-auto-check-trustdb** disables this option.

--use-agent

--no-use-agent
This is dummy option. **gpg2** always requires the agent.

--gpg-agent-info
This is dummy option. It has no effect when used with **gpg2**.

--agent-program *file*
Specify an agent program to be used for secret key operations. The default value is the '*/usr/bin/gpg-agent*'. This is only used as a fallback when the environment variable **GPG_AGENT_INFO** is not set or a running agent cannot be connected.

--lock-once
Lock the databases the first time a lock is requested and do not release the lock until the process terminates.

--lock-multiple
Release the locks every time a lock is no longer needed. Use this to override a previous **--lock-once** from a config file.

--lock-never
Disable locking entirely. This option should be used only in very special environments, where it can be assured that only one process is accessing those files. A bootable floppy with a stand-alone encryption system will probably use this. Improper usage of this option may lead to data and key corruption.

--exit-on-status-write-error
This option will cause write errors on the status FD to immediately terminate the process. That should in fact be the default but it never worked this way and thus we need an option to enable this, so that the change won't break applications which close their end of a status fd connected pipe too early. Using this option along with **--enable-progress-filter** may be used to cleanly cancel long running gpg operations.

--limit-card-insert-tries n
With **n** greater than 0 the number of prompts asking to insert a smartcard gets limited to N-1. Thus with a value of 1 gpg won't at all ask to insert a card if none has been inserted at startup. This option is useful in the configuration file in case an application does not know about the smartcard support and waits ad infinitum for an inserted card.

--no-random-seed-file
GnuPG uses a file to store its internal random pool over invocations. This makes random generation faster; however sometimes write operations are not desired. This option can be used to achieve that with the cost of slower random generation.

--no-greeting
Suppress the initial copyright message.

--no-secmem-warning
Suppress the warning about using insecure memory.

--no-permission-warning
Suppress the warning about unsafe file and home directory (**--homedir**) permissions. Note that the permission checks that GnuPG performs are not intended to be authoritative, but rather they simply warn about certain common permission problems. Do not assume that the lack of a warning means that your system is secure.

Note that the warning for unsafe **--homedir** permissions cannot be suppressed in the gpg.conf file, as this would allow an attacker to place an unsafe gpg.conf file in place, and use this file to suppress warnings about itself. The **--homedir** permissions warning may only be suppressed on the command line.

--no-mdc-warning
Suppress the warning about missing MDC integrity protection.

--require-secmem

--no-require-secmem
Refuse to run if GnuPG cannot get secure memory. Defaults to no (i.e. run, but give a warning).

--require-cross-certification

--no-require-cross-certification
When verifying a signature made from a subkey, ensure that the cross certification back signature on the subkey is present and valid. This protects against a subtle attack against subkeys that can sign. Defaults to **--require-cross-certification** for **gpg2**.

--expert

--no-expert
Allow the user to do certain nonsensical or silly things like signing an expired or revoked key, or certain potentially incompatible things like generating unusual key types. This also disables certain warning messages about potentially incompatible actions. As the name implies, this option is for experts only. If you don't fully understand the implications of what it allows you to do, leave this off. **--no-expert** disables this option.

Key related options

--recipient *name*

-r Encrypt for user id *name*. If this option or **--hidden-recipient** is not specified, GnuPG asks for the user-id unless **--default-recipient** is given.

--hidden-recipient *name*

-R Encrypt for user ID *name*, but hide the key ID of this user's key. This option helps to hide the receiver of the message and is a limited countermeasure against traffic analysis. If this option or **--recipient** is not specified, GnuPG asks for the user ID unless **--default-recipient** is given.

--encrypt-to name
Same as **--recipient** but this one is intended for use in the options file and may be used with your own user-id as an encrypt-to-self. These keys are only used when there are other recipients given either by use of **--recipient** or by the asked user id. No trust checking is performed for these user ids and even disabled keys can be used.

--hidden-encrypt-to name
Same as **--hidden-recipient** but this one is intended for use in the options file and may be used with your own user-id as a hidden encrypt-to-self. These keys are only used when there are other recipients given either by use of **--recipient** or by the asked user id. No trust checking is performed for these user ids and even disabled keys can be used.

--no-encrypt-to
Disable the use of all **--encrypt-to** and **--hidden-encrypt-to** keys.

--group name=value1
Sets up a named group, which is similar to aliases in email programs. Any time the group name is a recipient (**-r** or **--recipient**), it will be expanded to the values specified. Multiple groups with the same name are automatically merged into a single group.

The values are **key IDs**or fingerprin ts, but any key description is accepted. Note that a

value with spaces in it will be treated as two different values. Note also there is only one level of expansion --- you cannot make an group that points to another group. When used from the command line, it may be necessary to quote the argument to this option to prevent the shell from treating it as multiple arguments.

--ungroup name
Remove a given entry from the **--group** list.

--no-groups
Remove all entries from the **--group** list.

--local-user *name*

-u Use *name* as the key to sign with. Note that this option overrides **--default-key**.

--try-all-secrets
Don't look at the key ID as stored in the message but try all secret keys in turn to find the right decryption key. This option forces the behaviour as used by anonymous recipients (created by using **--throw-keyids** or **--hidden-recipient**) and might come handy in case where an encrypted message contains a bogus key ID.

--skip-hidden-recipients

--no-skip-hidden-recipients
During decryption skip all anonymous recipients. This option helps in the case that people use the hidden recipients feature to hide there own encrypt-to key from others. If oneself has many secret keys this may lead to a major annoyance because all keys are tried in turn to decrypt soemthing which was not really intended for it. The drawback of this option is that it is currently not possible to decrypt a message which includes real anonymous recipients.

Input and Output

--armor

-a Create ASCII armored output. The default is to create the binary OpenPGP format.

--no-armor
Assume the input data is not in ASCII armored format.

--output *file*

-o *file* Write output to *file*.

--max-output n
This option sets a limit on the number of bytes that will be generated when processing a file. Since OpenPGP supports various levels of compression, it is possible that the plaintext of a given message may be significantly larger than the original OpenPGP message. While GnuPG works properly with such messages, there is often a desire to set a maximum file size that will be generated before processing is forced to stop by the OS limits. Defaults to 0, which means no limit.

--import-options parameters
This is a space or comma delimited string that gives options for importing keys. Options can be prepended with a 'no-' to give the opposite meaning. The options are:

import-local-sigs
Allow importing key signatures marked as local. This is not generally useful unless a shared keyring scheme is being used. Defaults to no.

import-keep-ownertrust
Normally possible still existing ownertrust values of a key are cleared if a key is imported. This is in general desirable so that a formerly deleted key does not automatically gain an ownertrust values merely due to import. On the other hand it is sometimes necessary to re-import a trusted set of keys again but keeping already assigned ownertrust values. This can be achived by using this option.

repair-pks-subkey-bug
During import, attempt to repair the damage caused by the PKS keyserver bug (pre version 0.9.6) that mangles keys with multiple subkeys. Note that this cannot completely repair the damaged key as some crucial data is removed by the keyserver, but it does at least give you back one subkey. Defaults to no for regular **--import** and to yes for keyserver **--recv-keys**.

merge-only
During import, allow key updates to existing keys, but do not allow any new keys to be imported. Defaults to no.

import-clean
After import, compact (remove all signatures except the self-signature) any user IDs from the new key that are not usable. Then, remove any signatures from the new key that are not usable. This includes signatures that were issued by keys that are not present on the keyring. This option is the same as running the **--edit-key** command clean after import. Defaults to no.

import-minimal
Import the smallest key possible. This removes all signatures except the most recent self-signature on each user ID. This option is the same as running the **--edit-key** command minimize after import. Defaults to no.

--export-options parameters
This is a space or comma delimited string that gives options for exporting keys. Options can be prepended with a 'no-' to give the opposite meaning. The options are:

export-local-sigs
Allow exporting key signatures marked as local. This is not generally useful unless a shared keyring scheme is being used. Defaults to no.

export-attributes
Include attribute user IDs (photo IDs) while exporting. This is useful to export keys if they are going to be used by an OpenPGP program that does not accept attribute user IDs. Defaults to yes.

export-sensitive-revkeys
Include designated revoker information that was marked as sensitive. Defaults to no.

export-reset-subkey-passwd
When using the **--export-secret-subkeys** command, this option resets the passphrases for all exported subkeys to empty. This is useful when the exported subkey is to be used on an unattended machine where a passphrase doesn't

necessarily make sense. Defaults to no.

export-clean
Compact (remove all signatures from) user IDs on the key being exported if the user IDs are not usable. Also, do not export any signatures that are not usable. This includes signatures that were issued by keys that are not present on the keyring. This option is the same as running the **--edit-key** command clean before export except that the local copy of the key is not modified. Defaults to no.

export-minimal
Export the smallest key possible. This removes all signatures except the most recent self-signature on each user ID. This option is the same as running the **--edit-key** command minimize before export except that the local copy of the key is not modified. Defaults to no.

--with-colons
Print key listings delimited by colons. Note that the output will be encoded in UTF-8 regardless of any **--display-charset** setting. This format is useful when GnuPG is called from scripts and other programs as it is easily machine parsed. The details of this format are documented in the file '*doc/DETAILS*', which is included in the GnuPG source distribution.

--fixed-list-mode
Do not merge primary user ID and primary key in **--with-colon** listing mode and print all timestamps as seconds since 1970-01-01. Since GnuPG 2.0.10, this mode is always used and thus this option is obsolete; it does not harm to use it though.

--with-fingerprint
Same as the command **--fingerprint** but changes only the format of the output and may be used together with another command.

OpenPGP protocol specific options.

-t, --textmode

--no-textmode
Treat input files as text and store them in the OpenPGP canonical text form with standard CRLF line endings. This also sets the necessary flags to inform the recipient that the encrypted or signed data is text and may need its line endings converted back to whatever the local system uses. This option is useful when communicating between two platforms that have different line ending conventions (UNIX-like to Mac, Mac to Windows, etc). **--no-textmode** disables this option, and is the default.

--force-v3-sigs

--no-force-v3-sigs
OpenPGP states that an implementation should generate v4 signatures but PGP versions 5 through 7 only recognize v4 signatures on key material. This option forces v3 signatures for signatures on data. Note that this option implies **--no-ask-sig-expire**, and unsets **--sig-policy-url**, **--sig-notation**, and **--sig-keyserver-url**, as these features cannot be used with v3 signatures. **--no-force-v3-sigs** disables this option. Defaults to no.

--force-v4-certs

--no-force-v4-certs
Always use v4 key signatures even on v3 keys. This option also changes the default hash algorithm for v3 RSA keys from MD5 to SHA-1. **--no-force-v4-certs** disables this option.

--force-mdc
Force the use of encryption with a modification detection code. This is always used with the newer ciphers (those with a blocksize greater than 64 bits), or if all of the recipient keys indicate MDC support in their feature flags.

--disable-mdc
Disable the use of the modification detection code. Note that by using this option, the encrypted message becomes vulnerable to a message modification attack.

--personal-cipher-preferences string
Set the list of personal cipher preferences to **string**. Use **gpg2 --version** to get a list of available algorithms, and use **none** to set no preference at all. This allows the user to safely override the algorithm chosen by the recipient key preferences, as GPG will only select an algorithm that is usable by all recipients. The most highly ranked cipher in this list is also used for the **--symmetric** encryption command.

--personal-digest-preferences string
Set the list of personal digest preferences to **string**. Use **gpg2 --version** to get a list of available algorithms, and use **none** to set no preference at all. This allows the user to safely override the algorithm chosen by the recipient key preferences, as GPG will only select an algorithm that is usable by all recipients. The most highly ranked digest algorithm in this list is also used when signing without encryption (e.g. **--clearsign** or **--sign**).

--personal-compress-preferences string
Set the list of personal compression preferences to **string**. Use**gpg2 --v ersion** to get a list of available algorithms, and use **none** to set no preference at all. This allows the user to safely override the algorithm chosen by the recipient key preferences, as GPG will only select an algorithm that is usable by all recipients. The most highly ranked compression algorithm in this list is also used when there are no recipient keys to consider (e.g. **--symmetric**).

--s2k-cipher-algo name
Use **name** as the cipher algorithm used to protect secret keys. The default cipher is CAST5. This cipher is also used for conventional encryption if **--personal-cipher-preferences** and **--cipher-algo** is not given.

--s2k-digest-algo name
Use **name** as the digest algorithm used to mangle the passphrases. The default algorithm is SHA-1.

--s2k-mode n
Selects how passphrases are mangled. If **n** is 0 a plain passphrase (which is not recommended) will be used, a 1 adds a salt to the passphrase and a 3 (the default) iterates the whole process a number of times (see --s2k-count). Unless **--rfc1991** is used, this mode is also used for conventional encryption.

--s2k-count n
Specify how many times the passphrase mangling is repeated. This value may range between 1024 and 65011712 inclusive. The default is inquired from gpg-agent. Note that not all values in the 1024-65011712 range are legal and if an illegal value is selected, GnuPG will round up to the nearest legal value. This option is only meaningful if **--s2k-mode** is 3.

Compliance options

These options control what GnuPG is compliant to. Only one of these options may be active at a time. Note that the default setting of this is nearly always the correct one. See the INTEROPERABILITY WITH OTHER OPENPGP PROGRAMS section below before using one of these options.

--gnupg

Use standard GnuPG behavior. This is essentially OpenPGP behavior (see **--openpgp**), but with some additional workarounds for common compatibility problems in different versions of PGP. This is the default option, so it is not generally needed, but it may be useful to override a different compliance option in the gpg.conf file.

--openpgp

Reset all packet, cipher and digest options to strict OpenPGP behavior. Use this option to reset all previous options like **--s2k-***, **--cipher-algo**, **--digest-algo** and **--compress-algo** to OpenPGP compliant values. All PGP workarounds are disabled.

--rfc4880

Reset all packet, cipher and digest options to strict RFC-4880 behavior. Note that this is currently the same thing as **--openpgp**.

--rfc2440

Reset all packet, cipher and digest options to strict RFC-2440 behavior.

--rfc1991

Try to be more RFC-1991 (PGP 2.x) compliant.

--pgp2

Set up all options to be as PGP 2.x compliant as possible, and warn if an action is taken (e.g. encrypting to a non-RSA key) that will create a message that PGP 2.x will not be able to handle. Note that 'PGP 2.x' here means 'MIT PGP 2.6.2'. There are other versions of PGP 2.x available, but the MIT release is a good common baseline.

This option implies **--rfc1991 --disable-mdc --no-force-v4-certs --escape-from-lines --force-v3-sigs --allow-weak-digest-algos --cipher-algo IDEA --digest-algo MD5 --compress-algo ZIP**. It also disables**--textmode** when encrypting.

--pgp6

Set up all options to be as PGP 6 compliant as possible. This restricts you to the ciphers IDEA (if the IDEA plugin is installed), 3DES, and CAST5, the hashes MD5, SHA1 and RIPEMD160, and the compression algorithms none and ZIP. This also disables --throw-keyids, and making signatures with signing subkeys as PGP 6 does not understand signatures made by signing subkeys.

This option implies **--disable-mdc --escape-from-lines --force-v3-sigs**.

--pgp7

Set up all options to be as PGP 7 compliant as possible. This is identical to **--pgp6** except that MDCs are not disabled, and the list of allowable ciphers is expanded to add AES128, AES192, AES256, and TWOFISH.

--pgp8

Set up all options to be as PGP 8 compliant as possible. PGP 8 is a lot closer to the OpenPGP standard than previous versions of PGP, so all this does is disable **--throw-keyids** and set **--escape-from-lines**. All algorithms are allowed except for the SHA224, SHA384, and SHA512 digests.

Doing things one usually doesn't want to do.

-n

--dry-run
Don't make any changes (this is not completely implemented).

--list-only
Changes the behaviour of some commands. This is like **--dry-run** but different in some cases. The semantic of this command may be extended in the future. Currently it only skips the actual decryption pass and therefore enables a fast listing of the encryption keys.

-i

--interactive
Prompt before overwriting any files.

--debug-level *level*
Select the debug level for investigating problems. *level* may be a numeric value or by a keyword:

none No debugging at all. A value of less than 1 may be used instead of the keyword.

basic Some basic debug messages. A value between 1 and 2 may be used instead of the keyword.

advanced
More verbose debug messages. A value between 3 and 5 may be used instead of the keyword.

expert
Even more detailed messages. A value between 6 and 8 may be used instead of the keyword.

guru All of the debug messages you can get. A value greater than 8 may be used instead of the keyword. The creation of hash tracing files is only enabled if the keyword is used.

How these messages are mapped to the actual debugging flags is not specified and may change with newer releases of this program. They are however carefully selected to best aid in debugging.

--debug *flags*
Set debugging flags. All flags are or-ed and *flags* may be given in C syntax (e.g. 0x0042).

--debug-all
Set all useful debugging flags.

--faked-system-time *epoch*
This option is only useful for testing; it sets the system time back or forth to *epoch* which is the number of seconds elapsed since the year 1970. Alternatively *epoch* may be given as a full ISO time string (e.g. 20070924T154812).

--enable-progress-filter
Enable certain PROGRESS status outputs. This option allows frontends to display a progress indicator while gpg is processing larger files. There is a slight performance overhead using it.

--status-fd n
Write special status strings to the file descriptor **n**. See the file DETAILS in the documentation for a listing of them.

--status-file file
Same as **--status-fd**, except the status data is written to file **file**.

--logger-fd n
Write log output to file descriptor **n** and not to STDERR.

--log-file file

--logger-file file
Same as **--logger-fd**, except the logger data is written to file **file**. Note that **--log-file** is only implemented for GnuPG-2.

--attribute-fd n
Write attribute subpackets to the file descriptor **n**. This is most useful for use with **--status-fd**, since the status messages are needed to separate out the various subpackets from the stream delivered to the file descriptor.

--attribute-file file
Same as **--attribute-fd**, except the attribute data is written to file **file**.

--comment string

--no-comments
Use **string** as a comment string in clear text signatures and ASCII armored messages or keys (see **--armor**). The default behavior is not to use a comment string. **--comment** may be repeated multiple times to get multiple comment strings. **--no-comments** removes all comments. It is a good idea to keep the length of a single comment below 60 characters to avoid problems with mail programs wrapping such lines. Note that comment lines, like all other header lines, are not protected by the signature.

--emit-version

--no-emit-version
Force inclusion of the version string in ASCII armored output. If given once only the name of the program and the major number is emitted (default), given twice the minor is also emitted, given triple the micro is added, and given quad an operating system identification is also emitted. **--no-emit-version** disables the version line.

--sig-notation name=value

--cert-notation name=value

-N, --set-notation name=value
Put the name value pair into the signature as notation data. **name** m ust consist only of printable characters or spaces, and must contain a '@' character in the form keyname@domain.example.com (substituting the appropriate keyname and domain name, of course). This is to help prevent pollution of the IETF reserved notation namespace. The **--expert** flag overrides the '@' check. **value** may be any printable string; it will be encoded in UTF8, so you should check that your **--display-charset** is set correctly. If you prefix **name** with an exclamation mark (!), the notation data will be flagged as critical (rfc4880:5.2.3.16). **--sig-notation** sets a notation for data signatures. **--cert-notation** sets a notation for key signatures (certifications). **--set-notation** sets both.

There are special codes that may be used in notation names. %k will be expanded into the key ID of the key being signed, %K into the long key ID of the key being signed, %f

into the fingerprint of the key being signed, %s into the key ID of the key making the signature, %S into the long key ID of the key making the signature, %g into the fingerprint of the key making the signature (which might be a subkey), %p into the fingerprint of the primary key of the key making the signature, %c into the signature count from the OpenPGP smartcard, and %% results in a single %. %k, %K, and %f are only meaningful when making a key signature (certification), and %c is only meaningful when using the OpenPGP smartcard.

--sig-policy-url string

--cert-policy-url string

--set-policy-url string
Use **string** as a Policy URL for signatures (rfc4880:5.2.3.20). If you prefix it with an exclamation mark (!), the policy URL packet will be flagged as critical. **--sig-policy-url** sets a policy url for data signatures. **--cert-policy-url** sets a policy url for key signatures (certifications). **--set-policy-url** sets both.

The same %-expandos used for notation data are available here as well.

--sig-keyserver-url string
Use **string** as a preferred keyserver URL for data signatures. If you prefix it with an exclamation mark (!), the keyserver URL packet will be flagged as critical.

The same %-expandos used for notation data are available here as well.

--set-filename string
Use **string** as the filename which is stored inside messages. This overrides the default, which is to use the actual filename of the file being encrypted.

--for-your-eyes-only

--no-for-your-eyes-only
Set the 'for your eyes only' flag in the message. This causes GnuPG to refuse to save the file unless the **--output** option is given, and PGP to use a secure viewer with a claimed Tempest-resistant font to display the message. This option overrides **--set-filename**. **--no-for-your-eyes-only** disables this option.

--use-embedded-filename

--no-use-embedded-filename
Try to create a file with a name as embedded in the data. This can be a dangerous option as it allows to overwrite files. Defaults to no.

--cipher-algo name
Use **name** as cipher algorithm. Running the program with the command **--version** yields a list of supported algorithms. If this is not used the cipher algorithm is selected from the preferences stored with the key. In general, you do not want to use this option as it allows you to violate the OpenPGP standard. **--personal-cipher-preferences** is the safe w ay to accomplish the same thing.

--digest-algo name
Use **name** as the message digest algorithm. Running the program with the command **--version** yields a list of supported algorithms. In general, you do not want to use this option as it allows you to violate the OpenPGP standard. **--personal-digest-preferences** is the safe way to accomplish the same thing.

--compress-algo name
Use compression algorithm **name**. zlib is RFC-1950 ZLIB compression. zip is RFC-1951 ZIP compression which is used by PGP. bzip2 is a more modern compression scheme that can compress some things better than zip or zlib, but at the cost of more memory used during compression and decompression. uncompressed or none disables compression. If this option is not used, the default behavior is to examine the recipient key preferences to see which algorithms the recipient supports. If all else fails, ZIP is used for maximum compatibility.

ZLIB may give better compression results than ZIP, as the compression window size is not limited to 8k. BZIP2 may give even better compression results than that, but will use a significantly larger amount of memory while compressing and decompressing. This may be significant in low memory situations. Note, however, that PGP (all versions) only supports ZIP compression. Using any algorithm other than ZIP or none will make the message unreadable with PGP. In general, you do not want to use this option as it allows you to violate the OpenPGP standard. **--personal-compress-preferences** is the safe way to accomplish the same thing.

--cert-digest-algo name
Use **name** as the message digest algorithm used when signing a key. Running the program with the command **--version** yields a list of supported algorithms. Be aware that if you choose an algorithm that GnuPG supports but other OpenPGP implementations do not, then some users will not be able to use the key signatures you make, or quite possibly your entire key.

--disable-cipher-algo name
Never allow the use of **name** as cipher algorithm. The given name will not be checked so that a later loaded algorithm will still get disabled.

--disable-pubkey-algo name
Never allow the use of **name** as public key algorithm. The given name will not be checked so that a later loaded algorithm will still get disabled.

--throw-keyids

--no-throw-keyids
Do not put the recipient key IDs into encrypted messages. This helps to hide the receivers of the message and is a limited countermeasure against traffic analysis. ([Using a little social engineering anyone who is able to decrypt the message can check whether one of the other recipients is the one he suspects.]) On the receiving side, it may slow down the decryption process because all available secret keys must be tried. **--no-throw-keyids** disables this option. This option is essentially the same as using **--hidden-recipient** for all recipients.

--not-dash-escaped
This option changes the behavior of cleartext signatures so that they can be used for patch files. You should not send such an armored file via email because all spaces and line endings are hashed too. You can not use this option for data which has 5 dashes at the beginning of a line, patch files don't have this. A special armor header line tells GnuPG about this cleartext signature option.

--escape-from-lines

--no-escape-from-lines
Because some mailers change lines starting with From to>F rom it is good to handle such lines in a special way when creating cleartext signatures to prevent the mail system from breaking the signature. Note that all other PGP versions do it this way too. Enabled

by default. **--no-escape-from-lines** disables this option.

--passphrase-repeat n
Specify how many times **gpg2** will request a new passphrase be repeated. This is useful for helping memorize a passphrase. Defaults to 1 repetition.

--passphrase-fd n
Read the passphrase from file descriptor **n**. Only the first line will be read from file descriptor **n**. If you use 0 for **n**, the passphrase will be read from STDIN. This can only be used if only one passphrase is supplied. Note that this passphrase is only used if the option **--batch** has also been given. This is different from **gpg**.

--passphrase-file file
Read the passphrase from file **file**. Only the first line will be read from file **file**. This can only be used if only one passphrase is supplied. Obviously, a passphrase stored in a file is of questionable security if other users can read this file. Don't use this option if you can avoid it. Note that this passphrase is only used if the option **--batch** has also been given. This is different from **gpg**.

--passphrase string
Use **string** as the passphrase. This can only be used if only one passphrase is supplied. Obviously, this is of very questionable security on a multi-user system. Don't use this option if you can avoid it. Note that this passphrase is only used if the option **--batch** has also been given. This is different from **gpg**.

--command-fd n
This is a replacement for the deprecated shared-memory IPC mode. If this option is enabled, user input on questions is not expected from the TTY but from the given file descriptor. It should be used together with **--status-fd**. See the file doc/DETAILS in the source distribution for details on how to use it.

--command-file file
Same as **--command-fd**, except the commands are read out of file **file**

--allow-non-selfsigned-uid

--no-allow-non-selfsigned-uid
Allow the import and use of keys with user IDs which are not self-signed. This is not recommended, as a non self-signed user ID is trivial to forge. **--no-allow-non-selfsigned-uid** disables.

--allow-freeform-uid
Disable all checks on the form of the user ID while generating a new one. This option should only be used in very special environments as it does not ensure the de-facto standard format of user IDs.

--ignore-time-conflict
GnuPG normally checks that the timestamps associated with keys and signatures have plausible values. However, sometimes a signature seems to be older than the key due to clock problems. This option makes these checks just a warning. See also **--ignore-valid-from** for timestamp issues on subkeys.

--ignore-valid-from
GnuPG normally does not select and use subkeys created in the future. This option allows the use of such keys and thus exhibits the pre-1.0.7 behaviour. You should not use this option unless there is some clock problem. See also **--ignore-time-conflict** for

timestamp issues with signatures.

--ignore-crc-error
The ASCII armor used by OpenPGP is protected by a CRC checksum against transmission errors. Occasionally the CRC gets mangled somewhere on the transmission channel but the actual content (which is protected by the OpenPGP protocol anyway) is still okay. This option allows GnuPG to ignore CRC errors.

--ignore-mdc-error
This option changes a MDC integrity protection failure into a warning. This can be useful if a message is partially corrupt, but it is necessary to get as much data as possible out of the corrupt message. However, be aware that a MDC protection failure may also mean that the message was tampered with intentionally by an attacker.

--allow-weak-digest-algos
Signatures made with the broken MD5 algorithm are normally rejected with an ‘‘invalid digest algorithm’’ message. This option allows the verification of signatures made with such weak algorithms.

--no-default-keyring
Do not add the default keyrings to the list of keyrings. Note that GnuPG will not operate without any keyrings, so if you use this option and do not provide alternate keyrings via **--keyring** or **--secret-keyring**, then GnuPG will still use the default public or secret keyrings.

--skip-verify
Skip the signature verification step. This may be used to make the decryption faster if the signature verification is not needed.

--with-key-data
Print key listings delimited by colons (like **--with-colons**) and print the public key data.

--fast-list-mode
Changes the output of the list commands to work faster; this is achieved by leaving some parts empty. Some applications don’t need the user ID and the trust information given in the listings. By using this options they can get a faster listing. The exact behaviour of this option may change in future versions. If you are missing some information, don’t use this option.

--no-literal
This is not for normal use. Use the source to see for what it might be useful.

--set-filesize
This is not for normal use. Use the source to see for what it might be useful.

--show-session-key
Display the session key used for one message. See **--override-session-key** for the counterpart of this option.

We think that Key Escrow is a Bad Thing; however the user should have the freedom to decide whether to go to prison or to reveal the content of one specific message without compromising all messages ever encrypted for one secret key. DON’T USE IT UNLESS YOU ARE REALLY FORCED TO DO SO.

--override-session-key string
Don't use the public key but the session key **string**. The format of this string is the same as the one printed by **--show-session-key**. This option is normally not used but comes handy in case someone forces you to reveal the content of an encrypted message; using this option you can do this without handing out the secret key.

--ask-sig-expire

--no-ask-sig-expire
When making a data signature, prompt for an expiration time. If this option is not specified, the expiration time set via **--default-sig-expire** is used. **--no-ask-sig-expire** disables this option.

--default-sig-expire
The default expiration time to use for signature expiration. Valid values are 0 for no expiration, a number followed by the letter d (for days), w (for weeks), m (for months), or y (for years) (for example 2m for two months, or 5y for five years), or an absolute date in the form YYYY-MM-DD. Defaults to 0.

--ask-cert-expire

--no-ask-cert-expire
When making a key signature, prompt for an expiration time. If this option is not specified, the expiration time set via **--default-cert-expire** is used. **--no-ask-cert-expire** disables this option.

--default-cert-expire
The default expiration time to use for key signature expiration. Valid values are 0 for no expiration, a number followed by the letter d (for days), w (for weeks), m (for months), or y (for years) (for example 2m for two months, or 5y for five years), or an absolute date in the form YYYY-MM-DD. Defaults to 0.

--allow-secret-key-import
This is an obsolete option and is not used anywhere.

--allow-multiple-messages

--no-allow-multiple-messages
Allow processing of multiple OpenPGP messages contained in a single file or stream. Some programs that call GPG are not prepared to deal with multiple messages being processed together, so this option defaults to no. Note that versions of GPG prior to 1.4.7 always allowed multiple messages.

Warning: Do not use this option unless you need it as a temporary workaround!

--enable-special-filenames
This options enables a mode in which filenames of the form '*-&n*', where n is a non-negative decimal number, refer to the file descriptor n and not to a file with that name.

--no-expensive-trust-checks
Experimental use only.

--preserve-permissions
Don't change the permissions of a secret keyring back to user read/write only. Use this option only if you really know what you are doing.

--default-preference-list string
Set the list of default preferences to **string**. This preference list is used for new keys and becomes the default for setpref in the edit menu.

--default-keyserver-url name
Set the default keyserver URL to **name**. This keyserver will be used as the keyserver URL when writing a new self-signature on a key, which includes key generation and changing preferences.

--list-config
Display various internal configuration parameters of GnuPG. This option is intended for external programs that call GnuPG to perform tasks, and is thus not generally useful. See the file '*doc/DETAILS*' in the source distribution for the details of which configuration items may be listed. **--list-config** is only usable with **--with-colons** set.

--gpgconf-list
This command is similar to **--list-config** but in general only internally used by the **gpgconf** tool.

--gpgconf-test
This is more or less dummy action. However it parses the configuration file and returns with failure if the configuration file would prevent **gpg** from startup. Thus it may be used to run a syntax check on the configuration file.

Deprecated options

--show-photos

--no-show-photos
Causes **--list-keys**, **--list-sigs**, **--list-public-keys**, **--list-secret-keys**, and verifying a signature to also display the photo ID attached to the key, if any. See also **--photo-viewer**. These options are deprecated. Use **--list-options** [**no-**]**show-photos** and/or **--verify-options** [**no-**]**show-photos** instead.

--show-keyring
Display the keyring name at the head of key listings to show which keyring a given key resides on. This option is deprecated: use **--list-options** [**no-**]**show-keyring** instead.

--always-trust
Identical to **--trust-model always**. This option is deprecated.

--show-notation

--no-show-notation
Show signature notations in the **--list-sigs** or **--check-sigs** listings as well as when verifying a signature with a notation in it. These options are deprecated. Use **--list-options** [**no-**]**show-notation** and/or **--verify-options** [**no-**]**show-notation** instead.

--show-policy-url

--no-show-policy-url
Show policy URLs in the **--list-sigs** or **--check-sigs** listings as well as when verifying a signature with a policy URL in it. These options are deprecated. Use **--list-options** [**no-**]**show-policy-url** and/or **--verify-options** [**no-**]**show-policy-url** instead.

EXAMPLES

gpg -se -r Bob file
sign and encrypt for user Bob

gpg --clearsign file
make a clear text signature

gpg -sb file
make a detached signature

gpg -u 0x12345678 -sb file
make a detached signature with the key 0x12345678

gpg --list-keys user_ID
show keys

gpg --fingerprint user_ID
show fingerprint

gpg --verify pgpfile

gpg --verify sigfile
Verify the signature of the file but do not output the data. The second form is used for detached signatures, where **sigfile** is the detached signature (either ASCII armored or binary) and are the signed data; if this is not given, the name of the file holding the signed data is constructed by cutting off the extension (.asc or .sig) of **sigfile** or by asking the user for the filename.

HOW TO SPECIFY A USER ID

There are different ways to specify a user ID to GnuPG. Some of them are only valid for **gpg** others are only good for **gpgsm**. Here is the entire list of ways to specify a key:

By key Id.
This format is deduced from the length of the string and its content or **0x** prefix. The key Id of an X.509 certificate are the low 64 bits of its SHA-1 fingerprint. The use of key Ids is just a shortcut, for all automated processing the fingerprint should be used.

When using **gpg** an exclamation mark (!) may be appended to force using the specified primary or secondary key and not to try and calculate which primary or secondary key to use.

The last four lines of the example give the key ID in their long form as internally used by the OpenPGP protocol. You can see the long key ID using the option **--with-colons**.

234567C4
0F34E556E
01347A56A
0xAB123456

234AABBCC34567C4
0F323456784E56EAB
01AB3FED1347A5612
0x234AABBCC34567C4

By fingerprint.
This format is deduced from the length of the string and its content or the **0x** prefix. Note, that only the 20 byte version fingerprint is available with **gpgsm** (i.e. the SHA-1 hash of the certificate).

When using **gpg** an exclamation mark (!) may be appended to force using the specified primary or secondary key and not to try and calculate which primary or secondary key to use.

The best way to specify a key Id is by using the fingerprint. This avoids any ambiguities in case that there are duplicated key IDs.

1234343434343434C434343434343434
123434343434343C34343434343734349A3434
0E123434343434343434EAB3484343434343434
0xE123434343434343434EAB3484343434343434

(**gpgsm** also accepts colons between each pair of hexadecimal digits because this is the de-facto standard on how to present X.509 fingerprints.)

By exact match on OpenPGP user ID.
This is denoted by a leading equal sign. It does not make sense for X.509 certificates.

=Heinrich Heine <heinrichh@uni-duesseldorf.de>

By exact match on an email address.
This is indicated by enclosing the email address in the usual way with left and right angles.

<heinrichh@uni-duesseldorf.de>

By word match.
All words must match exactly (not case sensitive) but can appear in any order in the user ID or a subjects name. Words are any sequences of letters, digits, the underscore and all characters with bit 7 set.

+Heinrich Heine duesseldorf

By exact match on the subject's DN.
This is indicated by a leading slash, directly followed by the RFC-2253 encoded DN of the subject. Note that you can't use the string printed by gpgsm --list-keys because that one as been reordered and modified for better readability; use --with-colons to print the raw (but standard escaped) RFC-2253 string

/CN=Heinrich Heine,O=Poets,L=Paris,C=FR

By exact match on the issuer's DN.
This is indicated by a leading hash mark, directly followed by a slash and then directly followed by the rfc2253 encoded DN of the issuer. This should return the Root cert of the issuer. See note above.

#/CN=Root Cert,O=Poets,L=Paris,C=FR

By exact match on serial number and issuer's DN.
This is indicated by a hash mark, followed by the hexadecimal representation of the serial number, then followed by a slash and the RFC-2253 encoded DN of the issuer. See note above.

#4F03/CN=Root Cert,O=Poets,L=Paris,C=FR

By keygrip
This is indicated by an ampersand followed by the 40 hex digits of a keygrip. **gpgsm** prints the keygrip when using the command **--dump-cert**. It does not yet work for OpenPGP keys.

&D75F22C3F86E355877348498CDC92BD21010A480

By substring match.
This is the default mode but applications may want to explicitly indicate this by putting the asterisk in front. Match is not case sensitive.

Heine
*Heine

Please note that we have reused the hash mark identifier which was used in old GnuPG versions to indicate the so called local-id. It is not anymore used and there should be no conflict when used with X.509 stuff.

Using the RFC-2253 format of DNs has the drawback that it is not possible to map them back to the original encoding, however we don't have to do this because our key database stores this encoding as meta data.

FILES

There are a few configuration files to control certain aspects of **gpg2**'s operation. Unless noted, they are expected in the current home directory (see: [option --homedir]).

gpg.conf
This is the standard configuration file read by **gpg2** on startup. It may contain any valid long option; the leading two dashes may not be entered and the option may not be abbreviated. This default name may be changed on the command line (see: [gpg-option --options]). You should backup this file.

Note that on larger installations, it is useful to put predefined files into the directory '*/etc/skel/.gnupg/*' so that newly created users start up with a working configuration. For existing users the a small helper script is provided to create these files (see: [addgnupghome]).

For internal purposes **gpg2** creates and maintains a few other files; They all live in in the current home directory (see: [option --homedir]). Only the **gpg2** may modify these files.

~/.gnupg/pubring.gpg
The public keyring. You should backup this file.

~/.gnupg/pubring.gpg.lock
The lock file for the public keyring.

~/.gnupg/secring.gpg
The secret keyring. You should backup this file.

~/.gnupg/trustdb.gpg
The trust database. There is no need to backup this file; it is better to backup the ownertrust values (see: [option --export-ownertrust]).

~/.gnupg/trustdb.gpg.lock
The lock file for the trust database.

~/.gnupg/random_seed
A file used to preserve the state of the internal random pool.

~/.gnupg/secring.gpg.lock
The lock file for the secret keyring.

/usr[/local]/share/gnupg/options.skel
The skeleton options file.

/usr[/local]/lib/gnupg/
Default location for extensions.

Operation is further controlled by a few environment variables:

HOME
Used to locate the default home directory.

GNUPGHOME
If set directory used instead of ~/.gnupg.

GPG_AGENT_INFO
Used to locate the gpg-agent. The value consists of 3 colon delimited fields: The first is the path to the Unix Domain Socket, the second the PID of the gpg-agent and the protocol version which should be set to 1. When starting the gpg-agent as described in its documentation, this variable is set to the correct value. The option **--gpg-agent-info** can be used to override it.

PINENTRY_USER_DATA
This value is passed via gpg-agent to pinentry. It is useful to convey extra information to a custom pinentry.

COLUMNS

LINES
Used to size some displays to the full size of the screen.

LANGUAGE
Apart from its use by GNU, it is used in the W32 version to override the language selection done through the Registry. If used and set to a valid and available language name (*langid*), the file with the translation is loaded from

gpgdir/gnupg.nls/*langid*.mo. Here *gpgdir* is the directory out of which the gpg binary has been loaded. If it can't be loaded the Registry is tried and as last resort the native Windows locale system is used.

BUGS

On older systems this program should be installed as setuid(root). This is necessary to lock memory pages. Locking memory pages prevents the operating system from writing memory pages (which may contain passphrases or other sensitive material) to disk. If you get no warning message about insecure memory your operating system supports locking without being root. The program drops root privileges as soon as locked memory is allocated.

Note also that some systems (especially laptops) have the ability to "suspend to disk" (also known as "safe sleep" or "hibernate"). This writes all memory to disk before going into a low power or even powered off mode. Unless measures are taken in the operating system to protect the saved memory, passphrases or other sensitive material may be recoverable from it later.

Before you report a bug you should first search the mailing list archives for similar problems and second check whether such a bug has already been reported to our bug tracker at http://bugs.gnupg.org

SEE ALSO

gpgv(1), **gpgsm(1)**, **gpg-agent(1)**

The full documentation for this tool is maintained as a Texinfo manual. If GnuPG and the info program are properly installed at your site, the command

info gnupg

should give you access to the complete manual including a menu structure and an index.